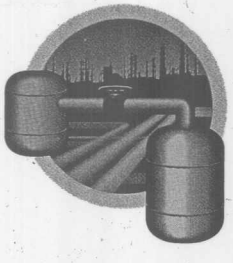
Loop Tuning Fundamentals

PID loop tuning may not be magic, but its intricacies do lie somewhere between science and art. The following proven tuning tips will help you craft your processes.

Vance J. VanDoren,
Control Engineering



AT A GLANCE

- PID explained
- Value selection tips
- Open- and closed-loop tuning
- Software product listing

A control loop is a feedback mechanism that attempts to correct discrepancies between a measured process variable and the desired setpoint. The controller applies the necessary corrective actions via an actuator that can drive the process variable up or down.

A proportional-integral-derivative (PID) controller tracks the error between the process variable and the setpoint, the integral of recent errors, and the rate by which the error has been changing. It computes its next corrective action from a weighted sum of those three terms (or modes), then outputs the results to the process and awaits the next measurement.

PID basics

A PID controller using the ideal or ISA standard form of the PID algorithm computes its output **CO(t)** according to the formula shown in Figure 1. **PV(t)** is the process variable measured at time **t** and the error **e(t)** is the difference between the process variable and the setpoint. The PID formula weights the proportional term by a factor of **P**, the integral term by a factor of $\mathbf{P/T_I}$, and the derivative term by a factor of $\mathbf{P \cdot T_D}$ where **P** is the controller gain, $\mathbf{T_I}$ is the integral time, and $\mathbf{T_D}$ is the derivative time.

This terminology bears some explaining. Gain refers to the amount by which the error signal will gain or lose strength as it passes through the controller en route to becoming part of the controller's output. A PID controller with a high gain will tend to generate aggressive corrective actions to eliminate errors.

The integral time refers to a hypothetical sequence of events where the error starts at zero then abruptly jumps to a fixed value. Such an error would cause an instantaneous response from the controller's proportional term and a response from the integral term that starts at zero and increases steadily. The time required for the integral term to catch up to the unchanging proportional term is the integral time $\mathbf{T_I}$. A PID controller with a long integral time is more heavily weighted towards proportional action than integral action.

Similarly, the derivative time $\mathbf{T_D}$ is a measure of the relative influence of the derivative term in the PID formula. If the error were to start at zero and begin increasing at a fixed rate, the proportional term would start at zero while the derivative term assumed a fixed value. The proportional term would then increase steadily until it caught up to the derivative term at the end of the derivative

The Ideal Form of the PID Formula

$$CO(t) = P\left[e(t) + \frac{1}{T_I}\left(\int e(t)\,dt\right) + T_D\left(\frac{d}{dt}e(t)\right)\right]$$

Proportional term | Integral term | Derivative term

Figure 1: *CO(t) is the controller's current output*
e(t) = SP - PV(t) is the error between the setpoint (SP) and the process variable PV(t)
P is the controller gain
T_I is the integral time
T_D is the derivative time
Source: Control Engineering

time. A PID controller with a long derivative time is more heavily weighted towards derivative action than proportional action.

Historical note

The very first feedback controllers included just the proportional term. For mathematical reasons that only became apparent later, a P-only controller tends to drive the error downward to a small but non-zero value and then quit. Operators observing this phenomenon would then manually increase the controller's output until the last vestiges of the error were eliminated. They called this operation resetting the controller.

When the integral term was introduced, operators observed that it tended to perform the reset operation automatically. That is, the controller would augment its proportional action just enough to eliminate the error entirely. Hence, integral action was originally called automatic reset and remains labeled that way on some PID controllers to this day. The derivative term was invented shortly thereafter and was described, accurately enough, as rate control.

Tricky business

Tuning is the art of selecting values for the tuning parameters P, T_I, and T_D so that the controller will be able to eliminate an error quickly without causing the process variable to fluctuate excessively. That's easier said than done.

Consider a car's cruise controller, for example. It can accelerate the car to a desired cruising speed, but not instantaneously. The car's inertia causes a delay between the time that the controller engages the accelerator and the time that the car's speed reaches the setpoint. How well a PID controller performs depends in large part on such lags.

Suppose an overloaded car with an undersized engine suddenly starts up a steep hill. The ensuing error between the car's actual and desired speeds would cause the controller's derivative and proportional actions to kick in immediately. The controller would begin to accelerate the car, but only as fast as the lag allows.

After a while, the integral action would also begin to contribute to the controller's output and eventually come to dominate it (since the error decreases so slowly when the lag time is

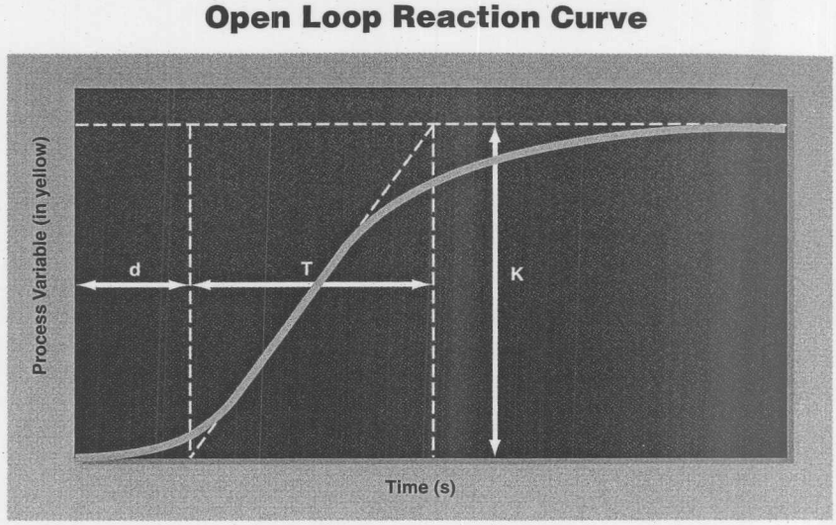


Figure 2: *An open-loop step test reveals the process' time constant **T**, dead-time **d**, and gain **K**.*
Source: Control Engineering

Help is available

Fortunately, there are commercial software products available that know all the tuning rules, all the different PID formulas, and all the latest tuning procedures. Exactly how each works is beyond the scope of this article and in many cases, proprietary. For more information, contact:

- **BESTune from BESTune.Com (www.bestune.isclever.com)**
- **Control Arts PID Tuning Software from Control Arts, Inc. (www.controlartsinc.com)**
- **Control Loop Assistant from Lambda Controls (www.lambdacontrols.com)**
- **Control Station from Control Station Technologies (www.controlstation.com)**
- **EnTech Tuner Module from the EnTech division of Emerson Process Management (www.entechcontrol.com)**
- **ExperTune* from ExperTune, Inc. (www.expertune.com)**
- **INTUNE* from ControlSoft, Inc. (www.controlsoftinc.com)**
- **pIDtune from EngineSoft (www.pidtune.com)**
- **Pitops from Artcon, Inc. (www.artcon.com)**
- **Protuner from Techmation (www.protuner.com)**
- **TOPAS from ACT, GmbH (www.act-control.com)**
- **Tune-Plus from Innoventon Industries, Inc. (www.innovin.com)**
- **TuneWizard from Plant Automation Services, Inc. (www.tunewizard.com)**

*These software packages are also available from several controller manufacturers as private-labeled products

long, and a sustained error is what drives the integral action). But exactly when that would happen and how dominant the integral action would become thereafter would depend on the severity of the lag and the relative sizes of the controller's integral and derivative times.

This simple example demonstrates a fundamental principle of PID tuning. The best choice for each of the tuning parameters **P**, T_I, and T_D depends on the values of the other two as well as the behavior of the controlled process. Furthermore, modifying the tuning of any one term affects the performance of the others since the modified controller affects the process and the process, in turn, affects the controller.

Ziegler-Nichols tuning

So how can a control engineer designing a PID loop determine the values for **P**, T_I, and T_D that will work best for a particular application? John G. Ziegler and Nathaniel B. Nichols of Taylor Instruments (now part of ABB Instrumentation in Rochester, NY) addressed that question in 1942 when they published two loop-tuning techniques that remain popular to this day.

Their open-loop technique is based on the results of a bump or step test for which the controller is taken off-line and manually forced to increase its output abruptly. A strip chart of the process variable's subsequent trajectory is known as the reaction curve (see Figure 2).

A sloped line drawn tangent to the reaction curve at its steepest point shows how fast the process reacted to the step change in the controller's output. The inverse of this line's slope is the process time constant **T** which measures the severity of the lag.

The reaction curve also shows how long it took for the process to demonstrate its initial reaction to the step (the dead time **d**) and how much the process variable increased relative to the size of the step (the process gain **K**). By trial-and-error, Ziegler and Nichols determined that the best settings for the tuning parameters **P**, T_I, and T_D could be computed from **T**, **d**, and **K** as follows:

$$P = \frac{1.2 \cdot T}{K \cdot d} \qquad T_I = 2.0 \cdot d \qquad T_D = 0.5 \cdot d$$

Once these parameter settings have been loaded into the PID formula and the controller returned to automatic mode, the controller should be able to eliminate future errors without causing the process variable to fluctuate excessively.

Ziegler and Nichols also described a closed loop-tuning technique that is conducted with the controller in automatic mode, but with the integral and derivative actions shut off. The controller gain is increased until even the slightest error causes a sustained oscillation in the process variable (see Figure 3).

The smallest controller gain that can cause such an oscillation is called the ultimate gain P_u. The period of those oscillations is called the ultimate period T_u. The appropriate tuning parameters can be computed from these two values according to the following rules:

$$P = 0.6 \cdot P_u \qquad T_I = 0.5 \cdot T_u \qquad T_D = 0.125 \cdot T_u$$

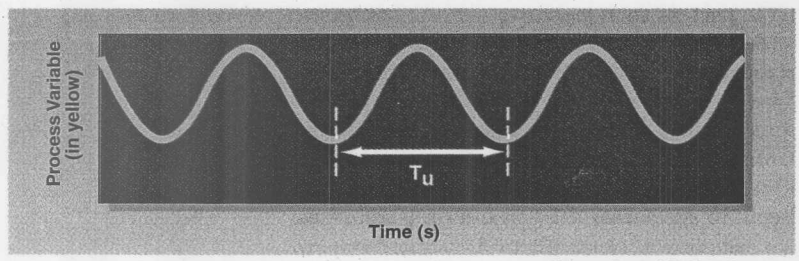
Caveats

Unfortunately, PID loop tuning isn't really that simple. Different PID controllers use different versions of the PID formula, and each must be tuned according to the appropriate set of rules. The rules also change when:

- The derivative and/or the integral action are disabled.
- The process itself is inherently oscillatory.
- The process behaves as if it contains its own integral term (as is the case with level control).
- The deadtime **d** is very small or significantly larger than the time constant **T**.

Furthermore, Ziegler and Nichols had a particular closed-loop performance objective in mind when they settled on their particular tuning rules. They chose to allow some fluctuations in the process variable so long as each successive peak was no more than one-fourth the size of its predecessor (so-called quarter wave decay). For applications that require even less fluctuation, additional tweaking of the tuning parameters is required.

This is where loop tuning becomes an art. It takes more than a little experience and sometimes a lot of luck to come up with just the right combination of **P**, T_I, and T_D. **ce**



Source: Control Engineering

Figure 3: *Forcing the closed-loop system into sustained oscillations with a proportional-only controller reveals the ultimate gain P_u and the ultimate period T_u.*